UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/932,578	08/17/2001	Blake Lewis	103.1072.01	5197

48102 7590 05/31/2007
NETWORK APPLIANCE/BLAKELY
12400 WILSHIRE BLVD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
LE, MIRANDA	
ART UNIT	**PAPER NUMBER**
2167	
MAIL DATE	**DELIVERY MODE**
05/31/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/932,578	LEWIS ET AL.	
	Examiner	Art Unit	
	Miranda Le	2167	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 April 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *25-35 and 40-55* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *25-35, 40-55* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 04/10/07 has been entered.

2. This communication is responsive to Amendment, filed 04/10/07.

Claims 25-35, 40-55 are pending in this application. In the amendment, claims 25, 35, 41, 43, 46-49 have been amended, claims 50-55 have been added. This action is made non-Final.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly

owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

4. Claims 32-34, 51, 53, 54, 55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Czajkowski (US Patent No. 6,453,403), in view of Hitz et al. (US Patent No. 5,819,292).

As per claim 32, Czajkowski teaches a method comprising:

maintaining an active map of information indicating in-use and free blocks associated with a file system *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3);*

maintaining a set of snapshots *(i.e. A series of snapshots, col. 7, lines 18-38; See Fig. 3)*, each snapshot representing a state of said active file system at a particular point in time, each snapshot having a corresponding active map indicating in-use blocks *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, col. 7, lines 18-38; See Fig. 3)* and free blocks *(i.e. the white memory blocks represent unallocated, or free, memory blocks , col. 7, lines 18-38; See Fig. 3)* of the active file system for a point in time *(i.e. a memory block pointed to by a "current" pointer, col. 3, lines 30-50)* at which said snapshot was generated *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has*

associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3); and

maintaining a summary map based on an active map included in at least one of said snapshot *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3);*

indicating said block is free in said summary map depending on a snapshot just prior to said particular snapshot and a snapshot just after said particular snapshot *(i.e. Fig. 3 show the block 321 is unused just after Heap Snapshot 304).*

Czajkowski does not specifically teach:

receiving a request to delete a particular snapshot; and

deleting said particular snapshot, wherein said deleting involves, for a block used by said particular snapshot.

Hitz teaches:

receiving a request to delete a particular snapshot *(i.e. Also, in step 530, the creation and deletion of snapshots, described below, are performed because it is the only point in time when*

the file system, except for the fsinfo block, is completely self consistent and about to be written to disk. A snapshot is deleted from the file system before a new one is created so that the same snapshot inode can be used in one pass, col. 13, lines 15-21);

deleting said particular snapshot, wherein said deleting involves, for a block used by said particular snapshot *(i.e. The sequential order of snapshots does not necessarily represent a chronological sequence of file system copies. Each individual snapshot in a file system can be deleted at any given time, thereby making an entry available for subsequent use. When BIT0 of a blkmap entry that references the active file system is cleared (indicating the block has been deleted from the active file system), the block cannot be reused if any of the snapshot reference bits are set. This is because the block is part of a snapshot that is still in use. A block can only be reused when all the bits in the blkmap entry are set to zero, col. 20, lines 52-62).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski and Hitz at the time the invention was made to modify the system of Czajkowski to include the limitations as taught by Hitz.

One of ordinary skill in the art would be motivated to make this combination in order to make an entry available for subsequent use in view of Hitz *(col. 20, lines 52-62)*, as doing so would give the added benefit of maintaining a consistent file system and creating read-only copies of the file system as taught by Hitz *(col. 1, lines 14-16).*

As per claim 51, Czajkowski teaches a method comprising:

maintaining a plurality of snapshots of a structured set of data in a data storage system, each snapshot representing a state of said structured set of data at a particular point in time, each

snapshot having associated therewith a separate active map indicating in-use blocks *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, col. 7, lines 18-38; See Fig. 3)* and free blocks *(i.e. the white memory blocks represent unallocated, or free, memory blocks , col. 7, lines 18-38; See Fig. 3)* of the structured set of data at the corresponding point in time *(i.e. a memory block pointed to by a "current" pointer, col. 3, lines 30-50; The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3);*

generating a summary map *(i.e. Heap Snapshot 306, Fig. 3)* which represents a summary of two or more of said active maps for different points in time *(i.e. a memory block pointed to by a "current" pointer, col. 3, lines 30-50),* by using a logical OR using a logical OR of the two or more of said active map *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3)*;

It should be noted that a logical OR has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, then both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

Czajkowski does not specifically teach:

making write allocation decisions relating to the structured set of data in the data storage system, based on the summary map, including using the summary map to avoid overwriting blocks used by a snapshot.

Hitz teaches making write allocation decisions relating to the structured set of data in the data storage system, based on the summary map *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37)*, including using the summary map to avoid overwriting blocks used by a snapshot *(i.e. The present invention prevents new data written to the active file system from overwriting "old" data that is part of a snapshot(s), col. 4, lines 33-44).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski and Hitz at the time the invention was made to modify the system of Czajkowski to include the limitations as taught by Hitz.

One of ordinary skill in the art would be motivated to make this combination in order to write every block that is part of the new consistency point to disk has disk space allocated for it *(col. 21, lines 43-50)* in view of Hitz, as doing so would give the added benefit of maintaining a consistent file system and creating read-only copies of the file system as taught by Hitz *(col. 1, lines 14-16).*

As per claim 33, Czajkowski teaches a method as in claim 32, wherein said indicating frees said block only when both:

said block is unused by said snapshot just prior to said particular snapshot *(i.e. Fig. 3 shown the block 321 is unused just prior to Heap Snapshot 304); and*

said block is unused by said snapshot just after said particular snapshot *(i.e. Fig. 3 show the block 321 is unused just after Heap Snapshot 304).*

As per claim 34, Hitz teaches a method as in claim 32, wherein said snapshot just after said particular snapshot corresponds to an active file system *(i.e. many different snapshots can be created for the same file system, col. 4, lines 21-32).*

As per claim 53, Czajkowski teaches a method as in claim 51, wherein the logical OR is an inclusive OR operation *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3)*;

It should be noted that an inclusive OR operation has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, then both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

As per claim 54, Czajkowski teaches a method as in claim 51, wherein said generating comprises performing a bitwise *(i.e. the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3)* logical operation on at least two of said active maps *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3).*

As per claim 55, Hitz teaches a method as in claim 51, wherein said making write allocation decisions *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37)* is further based on an active map indicating in-use blocks and free blocks associated with a current state of the structured set of data *(i.e. A first bit indicates whether a block is used by the active file system, and 20 remaining bits are used for up to 20 snapshots, however, some bits of the 31 bits may be used for other purposes, col. 4, lines 33-43).*

5. Claims 25-31, 35, 40, 49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Czajkowski (US Patent No. 6,453,403), in view of Lieuwen et al. (US Patent No. 6,272,502), and further in view of Hitz et al. (US Patent No. 5,819,292).

As per claim 25, Czajkowski teaches a method comprising:

maintaining an active map of information indicating in-use *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, col. 7, lines 18-38; See Fig. 3)* and free blocks *(i.e. the white memory blocks represent unallocated, or free, memory blocks , col. 7, lines 18-38; See Fig. 3)* associated with a file system *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3);*

maintaining a set of snapshots in the storage system *(i.e. A series of snapshots, col. 7, lines 18-38; See Fig. 3),* each snapshot representing a state of said file system at a particular point in time, each snapshot having a corresponding active map indicating in-use blocks *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, col. 7, lines 18-38; See Fig. 3)* and free blocks *(i.e. the white memory blocks represent unallocated, or free, memory blocks , col. 7, lines 18-38; See Fig. 3)* of the active file system for a point in time *(i.e. a memory block pointed to by a "current" pointer, col. 3, lines 30-50)* at which said snapshot was generated *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3); and*

computing a summary map *(i.e. See Heap Snapshot 306 in Fig. 3)* as a logical OR of the active maps *(i.e. See Block 317 and Block 318 of Heap Snapshots 304, 306 in Fig. 3);*

It should be noted that a logical OR has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

Czajkowski does not fairly teach:

a logical OR of the active maps of at least two of said snapshots.

Lieuwen teaches a logical OR of the active maps of at least two of said snapshots *(i.e. if any snapshot view in a viewgroup VG is updated in a transaction T, then all snapshot views in VG must be updated in T. Consequently, all snapshot views will be updated with the same periodicity, when updated, col. 3, lines 26-29).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski, Lieuwen at the time the invention was made to modify the system of Czajkowski to include the limitations as taught by Lieuwen.

One of ordinary skill in the art would be motivated to make this combination in order to update all snapshots views with the same periodicity in view of Lieuwen *(col. 3, lines 26-29),* as doing so would give the added benefit of having the system maintained data consistency within groups of the views, despite the different refreshing approaches taken for the different views in view of Lieuwen *(col. 1, lines 1-11).*

Czajkowski, Lieuwen do not explicitly teach:

using the summary map to make write allocation decisions in the storage system.

Hitz teaches using the summary map to make write allocation decisions in the storage system *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski, Lieuwen and Hitz at the time the invention was made to modify the system of Czajkowski, Lieuwen to include the limitations as taught by Hitz.

One of ordinary skill in the art would be motivated to make this combination in order to write every block that is part of the new consistency point to disk has disk space allocated for it

(col. 21, lines 43-50) in view of Hitz, as doing so would give the added benefit of maintaining a consistent file system and creating read-only copies of the file system as taught by Hitz *(col. 1, lines 14-16).*

As per claim 35, Czajkowski teaches a method comprising:

maintaining an active map of information indicating in-use *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, col. 7, lines 18-38; See Fig. 3)* and free blocks *(i.e. the white memory blocks represent unallocated, or free, memory blocks , col. 7, lines 18-38; See Fig. 3)* associated with a file system *(i.e. The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3);*

maintaining a set of snapshots, each snapshot representing a state of said file system at a particular point in time *(i.e. a memory block pointed to by a "current" pointer, col. 3, lines 30-50; The shaded blocks in the first four snapshots 302-308 represent allocated, or occupied, memory blocks, and the white memory blocks represent unallocated, or free, memory blocks. As mentioned above with reference to FIG. 2a, each memory block in the heap has associated with it a status bit which indicates the current allocation status of that block. In one embodiment, the*

status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3); and

maintaining a summary map *(i.e. Heap Snapshot 306, Fig. 3)* computed as a logical OR of the active maps *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3);*

It should be noted that a logical OR has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

Czajkowski does not specifically teach:

a logical OR of the active maps included in at least two of said snapshots.

Lieuwen teaches a logical OR of the active maps included in at least two of said snapshots *(i.e. if any snapshot view in a viewgroup VG is updated in a transaction T, then all snapshot views in VG must be updated in T. Consequently, all snapshot views will be updated with the same periodicity, when updated, col. 3, lines 26-29).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski, Lieuwen at the time the invention was made to modify the system of Czajkowski to include the limitations as taught by Lieuwen.

One of ordinary skill in the art would be motivated to make this combination in order to update all snapshots views with the same periodicity in view of Lieuwen *(col. 3, lines 26-29),* as doing so would give the added benefit of having the system maintained data consistency within groups of the views, despite the different refreshing approaches taken for the different views in view of Lieuwen *(col. 1, lines 1-11).*

Czajkowski, Lieuwen do not specifically teach:

selecting a set of blocks maintained by said file system for which to perform a write allocation operation;

updating only a portion of said summary map corresponding to said set of blocks, in response to said selecting; and

performing said write allocation operation in response to said updated summary map.

Hitz teaches:

selecting a set of blocks maintained by said file system for which to perform a write allocation operation *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37);*

updating only a portion of said summary map corresponding to said set of blocks, in response to said selecting *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37);*

performing said write allocation operation in response to said updated summary map *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski, Lieuwen and Hitz at the time the invention was made to modify the system of Czajkowski, Lieuwen to include the limitations as taught by Hitz.

One of ordinary skill in the art would be motivated to make this combination in order to write every block that is part of the new consistency point to disk has disk space allocated for it *(col. 21, lines 43-50)* in view of Hitz, as doing so would give the added benefit of maintaining a

consistent file system and creating read-only copies of the file system as taught by Hitz *(col. 1, lines 14-16).*

As per claim 26, Hitz teaches a method as in claim 25, further including:

making write allocation decisions in said file system based on said summary map *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37).*

As per claim 27, Czajkowski teaches a method as in claim 25, wherein said summary map is computed using an inclusive OR operation *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3).*

It should be noted that an inclusive OR operation has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, then both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

As per claim 28, Czajkowski teaches a method as in claim 25, wherein:

said set of snapshots includes at least two said snapshots *(See Fig. 3); and*

said computing includes performing a bitwise logical operation *(i.e. the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3)* on at least two said copies of earlier active maps included in said set of snapshots *(i.e. See Block 317 and Block 318*

in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3).

As per claim 29, Hitz teaches a method as in claim 25, wherein using the summary map to make write allocation decisions in the storage system comprises:

making write allocation decisions *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37)* based on both a current active map of the active file system and said summary map *(i.e. A first bit indicates whether a block is used by the active file system, and 20 remaining bits are used for up to 20 snapshots, however, some bits of the 31 bits may be used for other purposes, col. 4, lines 33-43).*

As per claim 30, Hitz teaches a method as in claim 25, wherein using the summary map to make write allocation decisions in the storage system comprises:

computing a combination of a current active map and said summary map *(i.e. A first bit indicates whether a block is used by the active file system, and 20 remaining bits are used for up to 20 snapshots, however, some bits of the 31 bits may be used for other purposes, col. 4, lines 33-43)*; and

making write allocation decisions based on a result of said computing *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37).*

As per claim 31, Hitz teaches a method as in claim 25, further including, for a selected portion of said summary map:

identifying a set of snapshots created since a recent update of said selected portion *(i.e. FIG. 18C is a diagram illustrating the active file system 1830 and a snapshot 1822 when a change to the active file system 1830 subsequently occurs after the snapshot 1822 is taken, col. 19, lines 19-41);* and

updating said selected portion based on only a most recent one of said snapshots *(i.e. As the active file system 1830 is modified in FIG. 18C, it uses more disk space because the file system comprising blocks 1812-1820 is not overwritten. In FIG. 18C, block 1818 is illustrated as a direct block. However, in an actual file system, block 1818 may be pointed to by indirect blocks as well. Thus, when block 1818 is modified and stored in a new disk location as block 1824, the corresponding direct and indirect blocks are also copied and assigned to the active file system 1830, col. 19, lines 42-50).*

As per claim 40, Czajkowski teaches a method as in claim 35, wherein said summary map is computed using an inclusive OR operation *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3).*

It should be noted that an inclusive OR operation has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, then both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

As per claim 49, Hitz teaches a method as in claim 25, wherein making write allocation decisions in said file system based on said summary map *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37)* comprises using the summary map to avoid overwriting blocks used by a snapshot *(i.e. The present invention prevents new data written to the active file system from overwriting "old" data that is part of a snapshot(s), col. 4, lines 33-44).*

6. Claims 43-48, 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hitz et al. (US Patent No. 5,819,292), in view of Czajkowski (US Patent No. 6,453,403), and further in view of Lieuwen et al. (US Patent No. 6,272,502).

As per claim 43, Hitz teaches a method comprising:

maintaining a plurality of persistent point-in-time images of a file system, each persistent point-in-time image representing a state of said file system at a particular point-in-time image representing a state of said file system at a particular point in time, each persistent point-in-time image having associated therewith a separate map indicating in-use blocks and free blocks of the file system at the corresponding point in time *(i.e. A block is available as a free block in the file system when all bits (BIT0-BIT31) in the 32-bit entry 1110A for the block are clear (reset to a value of 0). FIG. 11C is a diagram illustrating entry 1110A of FIG. 11A indicating the disk block is free. ... All the snapshot bits must also be zero for the block to be allocated. Bit 31 (BIT31) of entry 1110A always has the same state as bit 0 (BIT0) on disk, however, when loaded into memory bit 31 (BIT31) is used for bookkeeping as part of a consistency point, col. 9, line 55 to col. 10, line 18); and*

generating a summary map *(i.e. FIG. 11A indicating the disk block is free. Thus, the block referenced by entry 1110A of blkmap file 1110 is free when bits 0-31 (BIT0-BIT31) all have values of 0, col. 9, line 55 to col. 10, line 18).*

Hitz does not explicitly teach generating a summary map as a logical OR of said maps associated with at least two of said persistent point-in-time images.

Czajkowski teaches generating a summary map *(i.e. Heap Snapshot 306, Fig. 3)* as a logical OR of said maps *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3);*

It should be noted that a logical OR has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

It would have been obvious to one of ordinary skill of the art having the teaching of Hitz, Czajkowski at the time the invention was made to modify the system of Hitz to include the limitations as taught by Czajkowski.

One of ordinary skill in the art would be motivated to make this combination in order to use the Heap Snapshot for allocating the requested quantity of memory blocks *(col. 3, line 51 to col. 4, line 2)* in view of Czajkowski, as doing so would added the benefit of enabling any memory block whose corresponding structure or object has become unreferenced (i.e., is not "alive") to be reclaimed for future use as taught by Czajkowski *(col. 3, line 51 to col. 4, line 2).*

Lieuwen teaches a logical OR of said maps associated with at least two of said persistent point-in-time images *(i.e. if any snapshot view in a viewgroup VG is updated in a transaction T, then all snapshot views in VG must be updated in T. Consequently, all snapshot views will be*

updated with the same periodicity, when updated, col. 3, lines 26-29).

It would have been obvious to one of ordinary skill of the art having the teaching of Hitz, Czajkowski, Lieuwen at the time the invention was made to modify the system of Hitz, Czajkowski to include the limitations as taught by Lieuwen.

One of ordinary skill in the art would be motivated to make this combination in order to update all snapshots views with the same periodicity in view of Lieuwen *(col. 3, lines 26-29),* as doing so would give the added benefit of having the system maintained data consistency within groups of the views, despite the different refreshing approaches taken for the different views in view of Lieuwen *(col. 1, lines 1-11).*

As per claim 44, Hitz teaches a method as in claim 43, further including:

making write allocation decisions in said file system based on said summary map *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37).*

As per claim 45, Czajkowski teaches a method as in claim 44, wherein said summary map is computed using an inclusive OR operation *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3);*

It should be noted that an inclusive OR operation has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

As per claim 46, Czajkowski teaches a method as in claim 43, wherein:

said generating includes performing a bitwise logical operation *(i.e. the status bit of each memory block in the heap may be stored as a bit in a bitmap, such that the bitmap holds the status of each memory block in the heap, col. 7, lines 18-38; See Fig. 3)* on at least two of said maps associated with the plurality of persistent point-in time image *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3).*

As per claim 47, Hitz teaches a method of claim 43, further including:

making write *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37)* allocation decisions based on said summary map and a map indication in-used blocks and free blocks associated with a current state of the file system *(i.e. A first bit indicates whether a block is used by the active file system, and 20 remaining bits are used for up to 20 snapshots, however, some bits of the 31 bits may be used for other purposes, col. 4, lines 33-43).*

As per claim 48, Hitz teaches a method as in claim 43, further including:

determining a combination of said summary map and a map indicating in-use blocks and free blocks associated with a current state of the file system *(i.e. A first bit indicates whether a block is used by the active file system, and 20 remaining bits are used for up to 20 snapshots, however, some bits of the 31 bits may be used for other purposes, col. 4, lines 33-43); and*

making write allocation decisions based on a result of said determining *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37).*

As per claim 50, Hitz teaches a method as in claim 44, wherein making write allocation decisions in said file system based on said summary map *(i.e. all blocks of the blkmap file must be marked dirty here to ensure that step 730 write-allocates disk space for them, col. 21, lines 32-37)* comprises using the summary map to avoid overwriting blocks used by a snapshot *(i.e. The present invention prevents new data written to the active file system from overwriting "old" data that is part of a snapshot(s), col. 4, lines 33-44).*

7. Claims 41, 42 are rejected under 35 U.S.C. 103(a) as being unpatentable over Czajkowski (US Patent No. 6,453,403), in view of Hitz et al. (US Patent No. 5,819,292), and further in view of Lieuwen et al. (US Patent No. 6,272,502).

As per claim 41, Czajkowski teaches a method as in claim 32, wherein said summary map represents a logical OR of at least two copies of an earlier active map *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3).*

It should be noted that a logical OR operation has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, then both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

Czajkowski, Hitz do not teach a logical OR of at least two copies of an earlier active map included in at least two of said snapshots.

Lieuwen teaches a logical OR of at least two copies of an earlier active map included in at least two of said snapshots *(i.e. if any snapshot view in a viewgroup VG is updated in a transaction T, then all snapshot views in VG must be updated in T. Consequently, all snapshot views will be updated with the same periodicity, when updated, col. 3, lines 26-29).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski, Hitz, Lieuwen at the time the invention was made to modify the system of Czajkowski, Hitz to include the limitations as taught by Lieuwen.

One of ordinary skill in the art would be motivated to make this combination in order to update all snapshots views with the same periodicity in view of Lieuwen *(col. 3, lines 26-29),* as doing so would give the added benefit of having the system maintained data consistency within groups of the views, despite the different refreshing approaches taken for the different views in view of Lieuwen *(col. 1, lines 1-11).*

As per claim 42, Czajkowski teaches a method of claim 41, wherein said logical OR is an inclusive OR operation *(i.e. See Block 317 and Block 318 in Heap Snapshot 304 and Heap Snapshot 306 in Fig. 3).*

It should be noted that an inclusive OR operation has been applied to the two blocks 317, 318 form the Heap Snapshot 304 to compute the Heap Snapshot 306 such as either block 317 OR block 318 is un-free, then both blocks 317, 318 are marked as un-free as shown in Fig. 3, col. 7, lines 39-57.

8. Claim 52 is rejected under 35 U.S.C. 103(a) as being unpatentable over Czajkowski (US Patent No. 6,453,403), in view of Hitz et al. (US Patent No. 5,819,292), and further in view of Lieuwen et al. (US Patent No. 6,272,502), and further in view of Rungta (US Patent No. 6,484,186).

As per claim 52, Hitz teaches a method as in claim 51, wherein the data storage system comprises the structured set of data is a file system *(i.e. As the active file system 1830 is modified in FIG. 18C, it uses more disk space because the file system comprising blocks 1812-1820 is not overwritten. In FIG. 18C, block 1818 is illustrated as a direct block. However, in an actual file system, block 1818 may be pointed to by indirect blocks as well. Thus, when block 1818 is modified and stored in a new disk location as block 1824, the corresponding direct and indirect blocks are also copied and assigned to the active file system 1830, col. 19, lines 42-50).*

Czajkowski, Hitz do not explicitly teach a file server.

Rungta teaches a file server *(i.e. Computer system 105 further includes a snapshot unit ... or can archive files over a network to which computer system 105 is attached, e.g., to a server, col. 2, lines 28-39).*

It would have been obvious to one of ordinary skill of the art having the teaching of Czajkowski, Hitz, Rungta at the time the invention was made to modify the system of Czajkowski, Hitz to include the limitations as taught by Rungta.

One of ordinary skill in the art would be motivated to make this combination in order to archive files over a network to which computer system is attached to a server in view of Rungta *(col. 2, lines 28-39),* as doing so would give the added the benefit of allowing an archive

operation to back up a consistent version of files even while the files are open for writing as taught by Rungta *(col. 1, lines 36-39).*

Response to Arguments

9. Applicant's arguments regarding the prior arts do not suggest the newly amended claims with respect to claims 25-35, 40-55 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Miranda Le whose telephone number is (571) 272-4112. The examiner can normally be reached on Monday through Friday from 8:30 AM to 5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John R. Cottingham, can be reached on (571) 272-7079. The fax number to this Art Unit is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-3900.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Miranda Le
May 24, 2007